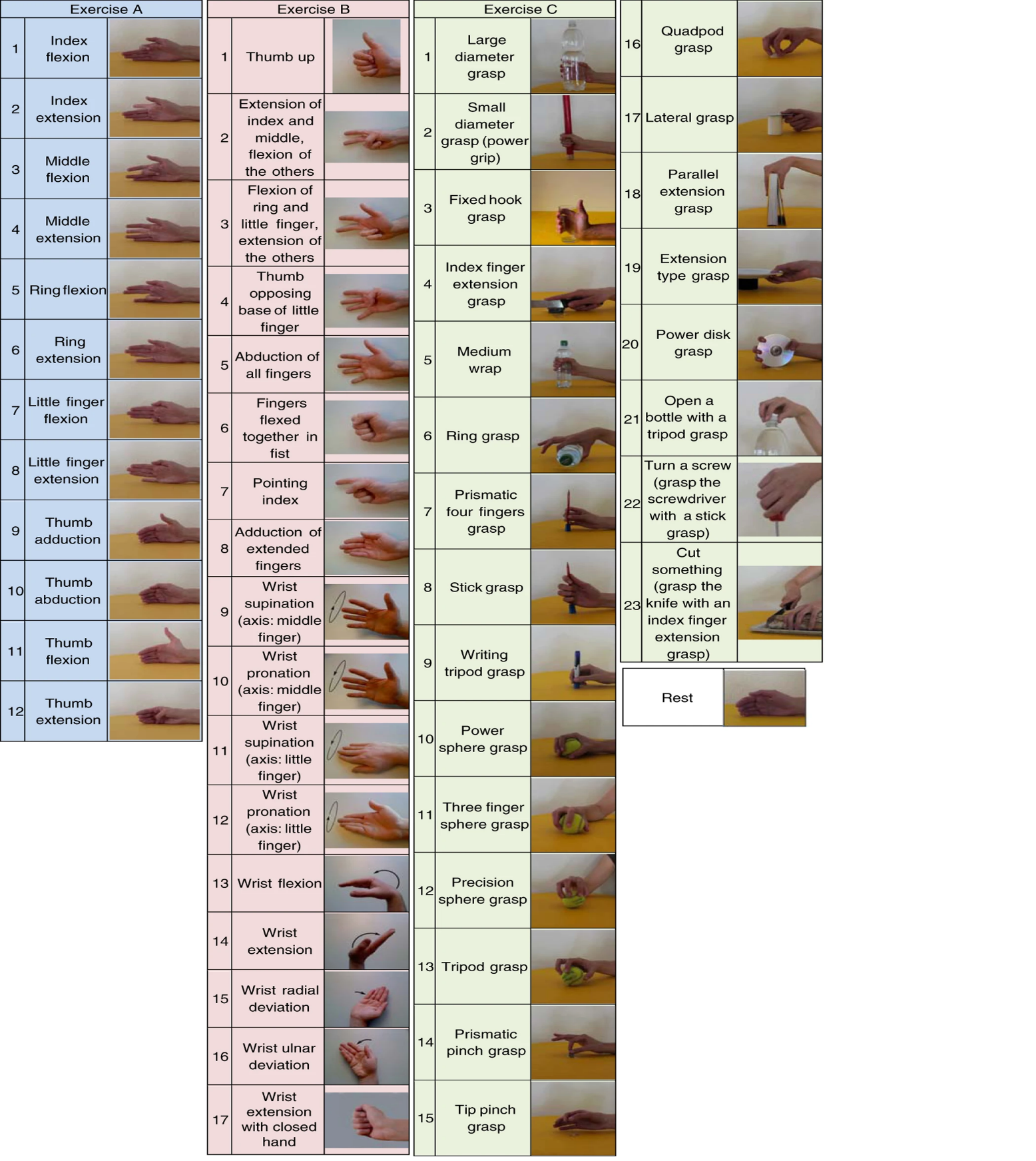
**Data Records**

<https://www.nature.com/articles/sdata201453/figures/2>

|  |  |
| --- | --- |
| Intact Subjects | 27 |
| Trans-radial Amputated Subjects | 0 |
| sEMG Electrodes | 10 Otto Bock |
| Total Number of Movements (rest included) | 53 |
| Number of Movement Repetitions | 10 |
| Reference in | Exercise A |
| Number of Movements | 12 |
| Ground Truth Parameter | Hand Kinematics |
| Hand Kinematics/Dynamic Sensors | Cyberglove II |
| Reference in | Exercise B |
| Number of Movements | 17 |
| Ground Truth Parameter | Hand Kinematics |
| Hand Kinematics/Dynamics Sensors | Cyberglove II |
| Reference in | Exercise C |
| Number of Movements | 23 |
| Ground Truth Parameter | Hand Kinematics |
| Hand Kinematics/Dynamics Sensors | Cyberglove II |



**Figure:** Movements and force patterns divided by exercise.

Exercise A (light blue): 12 basic movements of the fingers; Exercise B (red): 8 isometric and isotonic hand configurations and 9 basic movements of the wrist; Exercise C (green): 23 grasping and functional movements (everyday objects are presented to the subject for grasping, in order to mimic daily-life actions); Exercise D (purple): 9 force patterns; Rest position (white).

<https://www.nature.com/articles/sdata201453#Fig2>

Below is a description of the data sets' structure and content.

For each subject and exercise, the database contains one file in *Matlab* format ([www.mathworks.com](http://www.mathworks.com/)) with synchronized variables. The variables included in the files are:

* subject: subject number;
* exercise: exercise number;
* emg: Columns 1 through 8 show the signal from the electrodes evenly spaced around the forearm; columns 9 and 10 show the signal from the electrodes placed on the primary activity areas of the muscles Flexor Digitorum Superficialis and Extensor Digitorum Superficialis, respectively.
* glove (22 columns): uncalibrated signal from the 22 Cyberglove sensors. In the CyberGlove instructions, the raw data are stated to be proportional to the joint angles. [ninapro.hevs.ch/node/123](http://ninapro.hevs.ch/node/123);
* stimulus (1 column): the original label of the movement repeated by the subject;
* restimulus (1 column): the a-posteriori refined label of the movement;
* repetition (1 column): stimulus repetition index;
* rerepetition (1 column): restimulus repetition index;

<https://www.nature.com/articles/sdata201453#Fig2>

**Data Manipulation:**

For the classifications of 53 distinct motions, we used the 10 EMG value as an independent input variable, and all of the stimulus values were used as the dependent output variable. There are a total of 53 distinct stimulus values. We used the dataset 1 of the Nina Pro dataset to collect all the data from exercises A, B, and C.

Exercises A, B, and C each have between 0 and 12 distinct movements, 0 to 17 movements, and 0 to 23 motions, respectively.

Next, we take the data from Exercise A (1 to 12), Exercise B (1 to 17), Exercise C (1 to 23) and replace the value Exercise B (13 to 29) and Exercise C (30 to 52). Since the rest period for all 3 Exercise is 0, we leave it at 0.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Exercise A | | Exercise B | | Exercise C | |
| Given | **Transformed** | **Given** | **Transformed** | **Given** | **Transformed** |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 13 | 1 | 30 |
| 2 | 2 | 2 | 14 | 2 | 31 |
| 3 | 3 | 3 | 15 | 3 | 32 |
| 4 | 4 | 4 | 16 | 4 | 33 |
| 5 | 5 | 5 | 17 | 5 | 34 |
| 6 | 6 | 6 | 18 | 6 | 35 |
| 7 | 7 | 7 | 19 | 7 | 36 |
| 8 | 8 | 8 | 20 | 8 | 37 |
| 9 | 9 | 9 | 21 | 9 | 38 |
| 10 | 10 | 10 | 22 | 10 | 39 |
| 11 | 11 | 11 | 23 | 11 | 40 |
| 12 | 12 | 12 | 24 | 12 | 41 |
|  |  | 13 | 25 | 13 | 42 |
|  |  | 14 | 26 | 14 | 43 |
|  |  | 15 | 27 | 15 | 44 |
|  |  | 16 | 28 | 16 | 45 |
|  |  | 17 | 29 | 17 | 46 |
|  |  |  |  | 18 | 47 |
|  |  |  |  | 19 | 48 |
|  |  |  |  | 20 | 49 |
|  |  |  |  | 21 | 50 |
|  |  |  |  | 22 | 51 |
|  |  |  |  | 23 | 52 |

**Table**: Data Manipulation chart.

**Data Preprocessing:**

Since Nina Pro Dataset belongs to the big data category, it is not practical to run the entire data set at once since it would need a lot of processing time. Therefore, we test each participant from the 27 individuals individually. Next, we split the data into two groups: 90% for train data and 10% for test data. After that we use standard Scaler in order to scale the input variable. A common prerequisite for many machine learning estimators provided in scikit-learn is the standardization of datasets; if the individual features do not more or less resemble standard normally distributed data, they may perform poorly.

Standardize features by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as:

z = (x - u) / s

where u is the mean of the training samples or zero if with\_mean=False, and s is the standard deviation of the training samples or one if with\_std=False.

<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>

**Machine Learning Models:**

For our machine learning algorithm, we employed the Decision Tree Classifier, KNeighbors Classifier, Random Forest Classifier, Extra Trees Classifier, and XGBoost. To fine-tune the parameters, we utilized Grid Search CV with three levels of cross validation. The default settings for the Decision Tree Classifier, Random Forest Classifier, and Extra Trees Classifier will get the best results; for XGBoost, we choose "gpu predictor" as the predictor, and for KNeighbors Classifier, we choose "ball tree" as the method, with "n neighbors" equaling 2, "p" equaling 1, and "weights" equal to "distance." There was space for improvement by boosting the estimators for all tree-based methods, however this was impractical owing to a lack of computing configurations.

**Outlier Detection:**

We initially ran every required machine learning method to check the accuracy before looking for the outlier. We found the accuracy is low so the data contain outliers. Therefore, we picked those values where the stimulus and the restimulus values are the same in order to remove the outlier. Following the removal of the outliers, our models significantly improved. Here are the findings,

|  |  |  |
| --- | --- | --- |
| Algorithms | With Outliers | Without Outliers |
| Decision Tree Classifier | 66.394 | 86.131 |
| KNeighbors Classifier | 73.488 | 92.513 |
| Random Forest Classifier | 76.192 | 93.752 |
| Extra Trees Classifier | 76.383 | 93.871 |
| XGBoost | 73.703 | 90.162 |

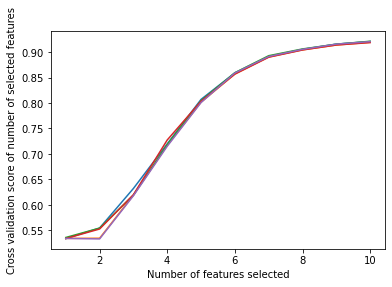
**Table**: Improvement comparison table with and without outliers

**Figure**: Bar plot of Improvement comparison with and without outliers.

**Feature Selection**

Recursive feature elimination with cross validation is the Wrapper Method that we used for Feature Selection. We select the Random Forest Classifier for estimator, step = 1, and Cross Validation = 5. The best number of features was then discovered to be 10, indicating that the entire value of the emg signal is required and cannot be reduced.

A curve of five cross-validation results is shown below, demonstrating how accuracy improves with each emg value.



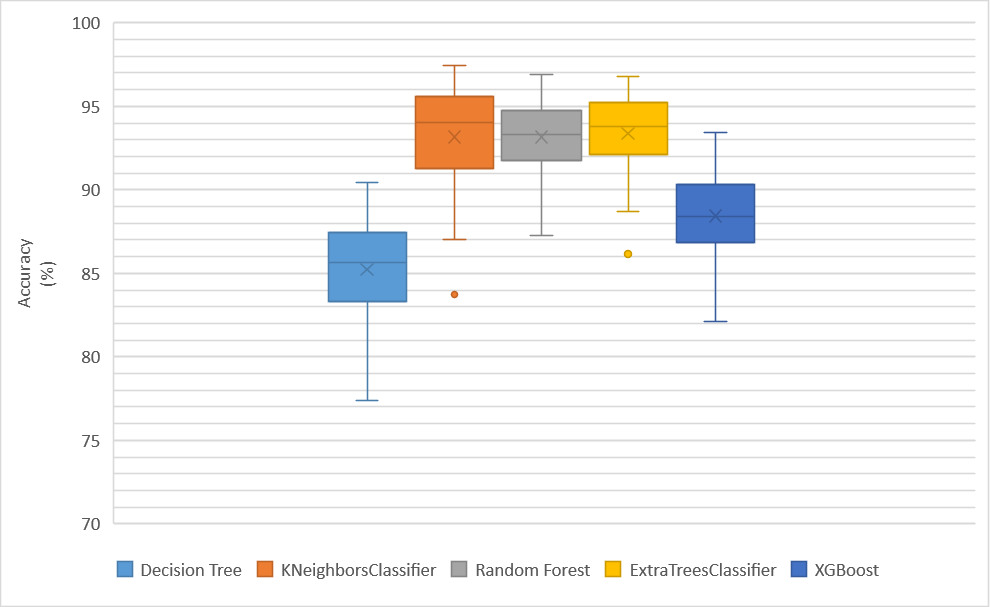
**Figure**: Optimal Number of features for Recursive feature elimination with each cross validation.

**Raw Machine Learning Results**

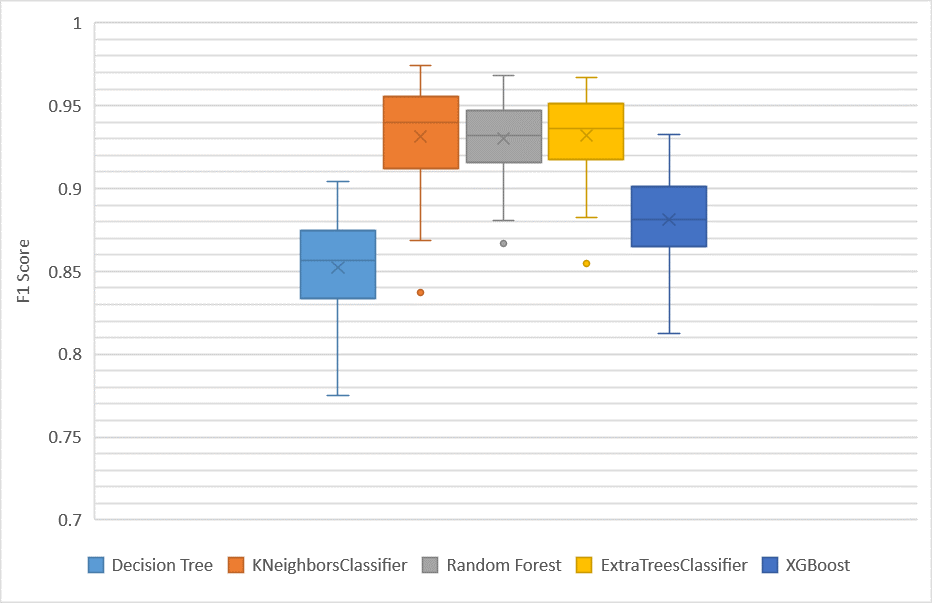
The following shows the outcomes of applying machine learning algorithms to all of the subjects in Ninapro dataset 1 after taking all of the EMG values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithms | Accuracy (%) | Precision | Recall | F1 Score |
| DT | 85.241 | 0.853 | 0.852 | 0.852 |
| KNN | 93.174 | 0.931 | 0.932 | 0.931 |
| RF | 93.177 | 0.932 | 0.932 | 0.930 |
| ET | 93.376 | 0.935 | 0.934 | 0.932 |
| XGB | 88.439 | 0.882 | 0.884 | 0.881 |

**Table:** Classification Results of Different Machine Learning Algorithms



**Figure:** Box Plot of Classification Accuracy Comparison of Different Machine Learning Algorithms



**Figure:** Box Plot of F1 Score Comparison of Different Machine Learning Algorithms

**Dimensionality Reduction:**

**T-distributed Stochastic Neighbor Embedding**

Following is the accuracy results of the t-distributed stochastic neighbor embedding’s individual elements.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithms | E1 | E2 | E3 |
| Decision Tree Classifier | 59.154 | 83.909 | 86.400 |
| KNeighbors Classifier | 65.911 | 86.672 | 88.604 |
| Random Forest Classifier | 59.143 | 87.037 | 89.716 |
| Extra Trees Classifier | 59.191 | 86.921 | 89.832 |
| XGBoost | 52.940 | 84.092 | 85.227 |

**Table**: Comparison table for every element of TSNE.

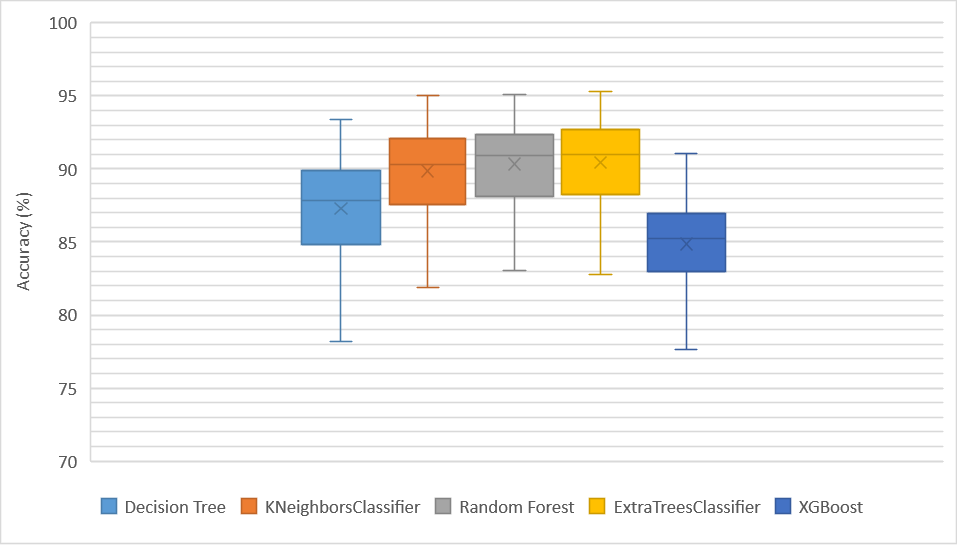
**Figure**: Bar Plot Comparison for every element of TSNE.

**Dimensionality Reduction: T-distributed Stochastic Neighbor Embedding (3 Element)**

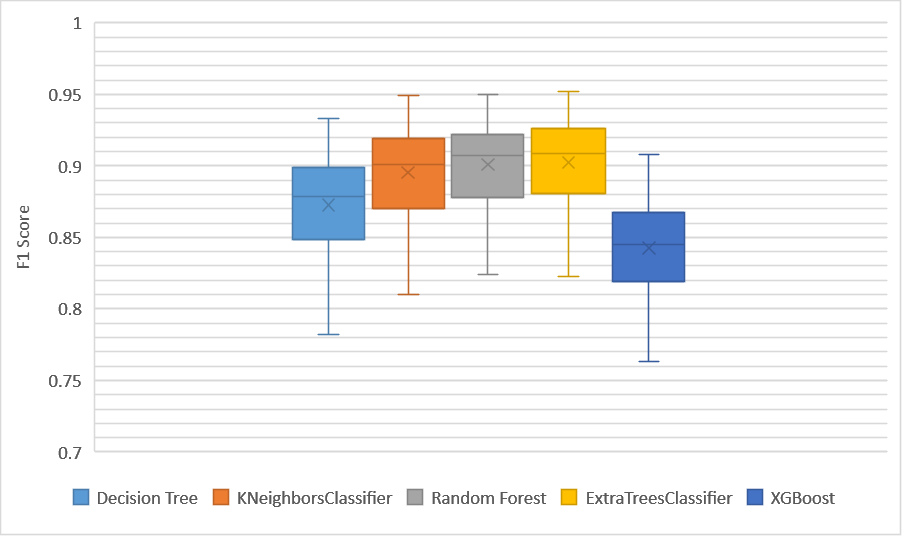
The following shows the outcomes of using TSNE with three components, followed by the application of machine learning algorithms to each subject in Ninapro dataset 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithms | Accuracy (%) | Precision | Recall | F1 Score |
| DT | 87.282 | 0.873 | 0.873 | 0.873 |
| KNN | 89.844 | 0.896 | 0.898 | 0.895 |
| RF | 90.328 | 0.901 | 0.903 | 0.901 |
| ET | 90.429 | 0.902 | 0.904 | 0.902 |
| XGB | 84.872 | 0.846 | 0.849 | 0.843 |

**Table:** Classification Results of Different Machine Learning Algorithms with TSNE as Dimensionality Reduction



**Figure:** Box Plot of Accuracy Comparison of Different Machine Learning Algorithms with TSNE as Dimensionality Reduction



**Figure:** Box Plot of F1 Score Comparison of Different Machine Learning Algorithms with TSNE as Dimensionality Reduction.

**Principal Component Analysis**

The accuracy results for each component of the principal component analysis are listed below.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithms | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 |
| DT | 57.857 | 57.804 | 57.862 | 57.838 | 57.883 | 57.846 | 57.746 | 81.579 | 81.466 | 81.513 |
| KNN | 64.228 | 64.228 | 64.228 | 64.228 | 64.228 | 64.228 | 64.228 | 89.887 | 89.887 | 89.887 |
| RF | 64.485 | 64.609 | 64.524 | 64.548 | 64.485 | 64.551 | 64.545 | 90.223 | 90.186 | 90.326 |
| ET | 64.342 | 64.397 | 64.249 | 64.387 | 64.284 | 64.255 | 64.405 | 90.804 | 90.825 | 90.883 |
| XGB | 64.870 | 64.870 | 64.870 | 64.870 | 64.870 | 64.870 | 64.870 | 85.167 | 85.167 | 85.167 |

**Table**: Comparison table for every element of PCA.

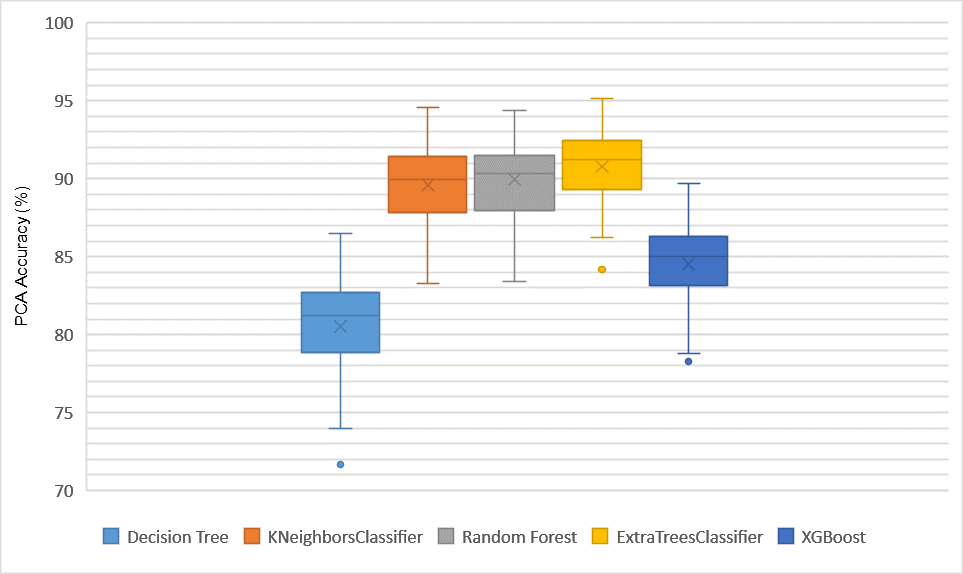
**Figure**: Bar Plot Comparison for every element of PCA.

**Dimensionality Reduction: Principal Component Analysis**

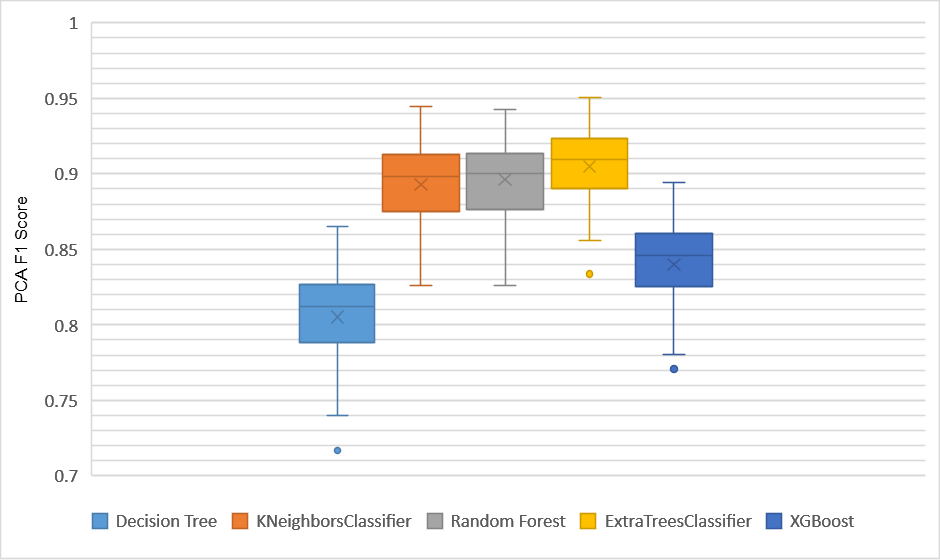
The results of applying machine learning techniques to each subject in Ninapro dataset 1 are shown in the section below, which uses Principal Component Analysis with eight components.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithms | Accuracy (%) | Precision | Recall | F1 Score |
| DT | 80.521 | 0.806 | 0.805 | 0.805 |
| KNN | 89.594 | 0.894 | 0.896 | 0.893 |
| RF | 89.949 | 0.899 | 0.899 | 0.896 |
| ET | 90.769 | 0.908 | 0.908 | 0.905 |
| XGB | 84.525 | 0.841 | 0.845 | 0.839 |

**Table:** Classification Results of Different Machine Learning Algorithms with PCA as Dimensionality Reduction



**Figure:** Box Plot of Accuracy Comparison of Different Machine Learning Algorithms with PCA as Dimensionality Reduction



**Figure:** Box Plot of F1 Score Comparison of Different Machine Learning Algorithms with PCA as Dimensionality Reduction.

**Independent Component Analysis**

Following is a list of the Independent Component Analysis accuracy results for each component.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithms | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 |
| DT | 53.059 | 54.071 | 60.403 | 68.088 | 73.847 | 79.133 | 81.104 | 82.847 | 83.809 | 83.978 |
| KNN | 51.884 | 55.939 | 64.067 | 73.757 | 80.657 | 85.729 | 88.680 | 89.599 | 90.519 | 90.524 |
| RF | 53.033 | 56.0496 | 64.934 | 75.052 | 82.379 | 87.589 | 90.263 | 91.465 | 92.556 | 92.651 |
| ET | 51.884 | 54.858 | 64.625 | 75.189 | 82.956 | 88.096 | 90.783 | 91.789 | 93.168 | 93.287 |
| XGB | 53.897 | 58.253 | 65.084 | 71.446 | 77.455 | 82.258 | 85.502 | 87.312 | 88.942 | 89.129 |

**Table**: Comparison table for every element of ICA.

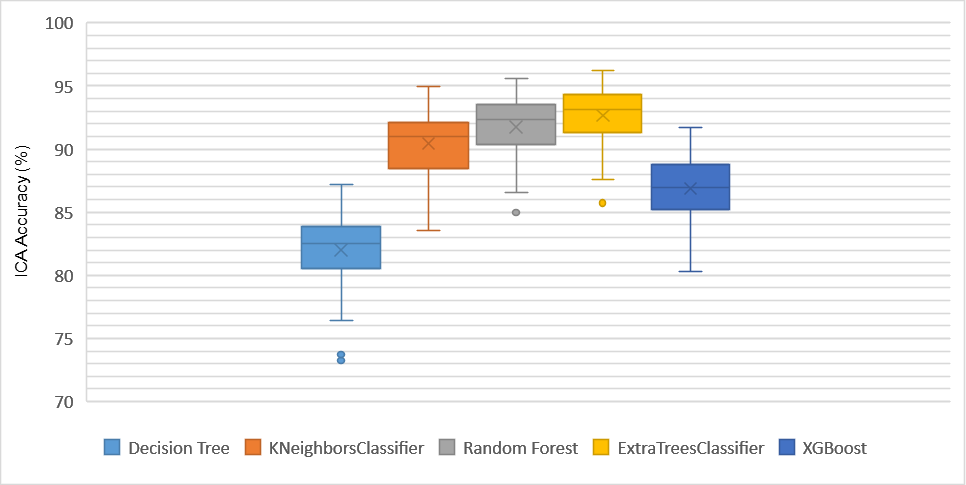
**Figure**: Bar Plot Comparison for every element of ICA.

**Dimensionality Reduction:** **Independent Component Analysis**

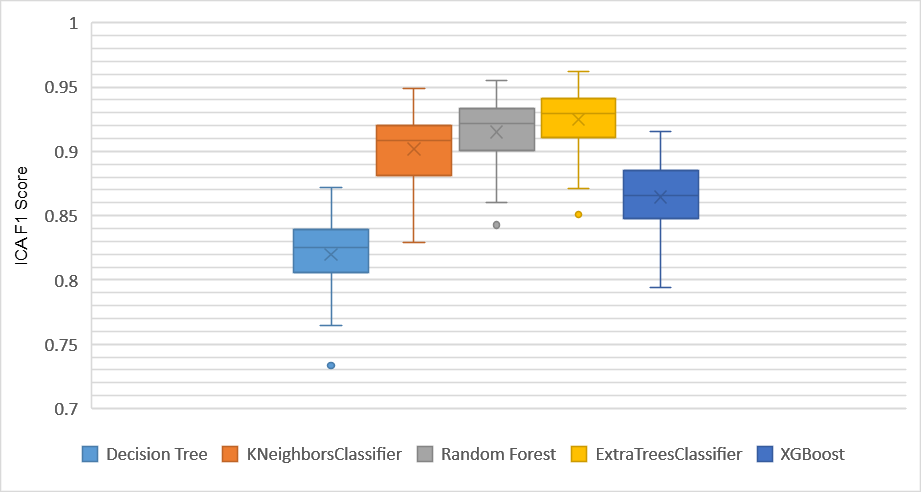
The section below uses Independent Component Analysis with Nine components to show the outcomes of applying machine learning algorithms to each subject in Ninapro dataset 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithms | Accuracy (%) | Precision | Recall | F1 Score |
| DT | 81.988 | 0.820 | 0.819 | 0.819 |
| KNN | 90.437 | 0.903 | 0.904 | 0.902 |
| RF | 91.734 | 0.917 | 0.917 | 0.915 |
| ET | 92.661 | 0.927 | 0.927 | 0.925 |
| XGB | 86.868 | 0.866 | 0.869 | 0.864 |

**Table:** Classification Results of Different Machine Learning Algorithms with ICA as Dimensionality Reduction

****

**Figure:** Box Plot of Accuracy Comparison of Different Machine Learning Algorithms with ICA as Dimensionality Reduction



**Figure:** Box Plot of F1 Score Comparison of Different Machine Learning Algorithms with ICA as Dimensionality Reduction.

**Comparison of raw data with various dimensionality reductions**

For the linear dimensionality reduction, we used PCA with 8 Elements, ICA with 9 Elements, and for the non-linear dimensionality reduction, TSNE with 3 Elements. We compared these methods to the raw data with all of the Elements.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithms | PCA (%)  (Element = 8) | ICA (%)  (Element = 9) | TSNE (%)  (Element = 3) | Raw Data (%)  (All Elements) |
| Decision Tree Classifier | 80.521 | 81.988 | 87.282 | 85.241 |
| KNeighbors Classifier | 89.594 | 90.437 | 89.844 | 93.174 |
| Random Forest | 89.949 | 91.734 | 90.328 | 93.177 |
| Extra Trees Classifier | 90.769 | 92.661 | 90.429 | 93.376 |
| XGBoost | 84.525 | 86.868 | 84.872 | 88.439 |

**Table:** Comparison of accuracy for various dimensionality reduction techniques with raw data.

**Figure:** Bar PlotComparison of accuracy for various dimensionality reduction techniques with raw data.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithms | PCA (%)  (Element = 8) | ICA (%)  (Element = 9) | TSNE (%)  (Element = 3) | Raw Data (%)  (All Elements) |
| Decision Tree Classifier | 0.805 | 0.819 | 0.873 | 0.852 |
| KNeighbors Classifier | 0.893 | 0.902 | 0.895 | 0.931 |
| Random Forest | 0.896 | 0.915 | 0.901 | 0.930 |
| Extra Trees Classifier | 0.905 | 0.925 | 0.902 | 0.932 |
| XGBoost | 0.839 | 0.864 | 0.843 | 0.881 |

**Table:** Comparison of F1 Score for various dimensionality reduction techniques with raw data

**Figure:** Bar PlotComparison of F1 Score for various dimensionality reduction techniques with raw data.

**Time Calculations:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithms | PCA  (Element = 8) | ICA  (Element = 9) | TSNE  (Element = 3) | Raw Data  (All Elements) |
| Decision Tree Classifier | 17.895 | 16.073 | 19.463 | 6.776 |
| KNeighbors Classifier | 2.415 | 3.002 | 0.079 | 2.492 |
| Random Forest | 296.319 | 356.119 | 552.169 | 169.559 |
| Extra Trees Classifier | 117.722 | 148.147 | 85.729 | 138.323 |
| XGBoost | 2516.577 | 3509.021 | 1530.200 | 1187.911 |

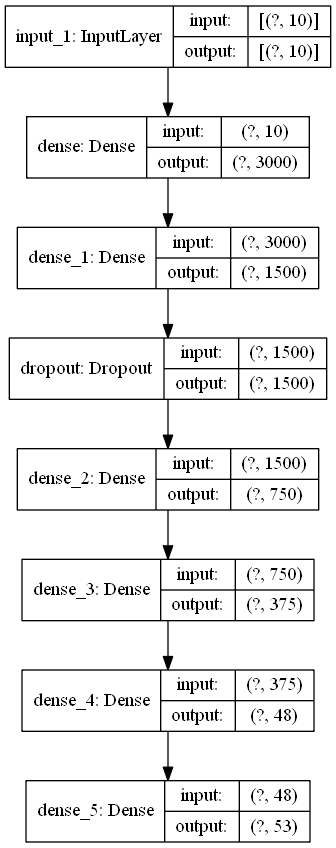
**Table:** Comparison of Time Calculations

**Deep Learning:**

We scaled the input data using a standard scaler and encoded the data with one hot encoding prior to applying the deep learning model.

We begin with 10 emg signals as inputs, then add a dense layer of 3000 neurons with a relu activation function, followed by another dense layer of 1500 neurons with a relu activation function. Next, a 0.2 dropout layer is added. 3 dense layers with activation functions of 750, 375, and 48 correspondingly are then added. A dense layer as 53 class with a SoftMax activation function is added last.

Then, with batch size set to 9000 and 300 epochs, we compile the model using Adam as the optimizer and categorical cross entropy as the loss function.



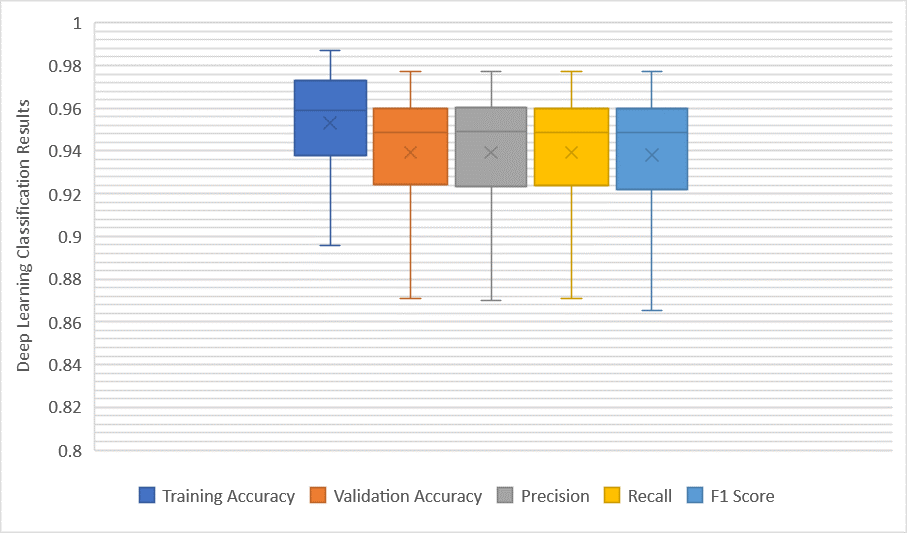
**Figure:** Deep Learning Architecture.

**Deep Learning Results**

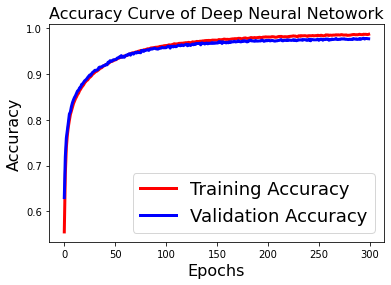
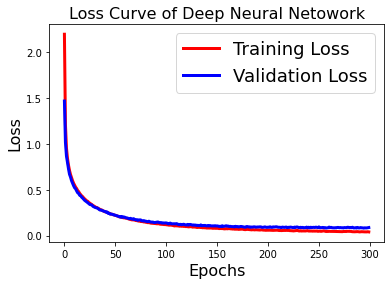
The following shows the outcomes of applying Deep learning algorithms to all of the subjects in Ninapro dataset 1 after taking all of the EMG values.

|  |  |
| --- | --- |
| **Training Accuracy** | 0.953204 |
| **Validation Accuracy** | 0.939196 |
| **Training Loss** | 0.148781 |
| **Validation Loss** | 0.209889 |
| **Precision** | 0.939182 |
| **Recall** | 0.9392 |
| **F1 Score** | 0.938018 |

**Table**: Deep Learning Mean Results of All 27 Subject of Dataset 1.



**Figure**: Box Plot of Deep Learning Classification Results of All 27 Subject of Dataset 1.



**Figure:** Accuracy Curve and Loss Curve of Subject 24.